Explicit Graph Reasoning Fusing Knowledge and Contextual Information for Multi-hop Question Answering

Zhenyun Deng, Yonghua Zhu, Qianqian Qi, Michael Witbrock, Patricia Riddle
School of Computer Science, University of Auckland, New Zealand
{zden658, yzhu970, qqi518}@aucklanduni.ac.nz
{m.witbrock, p.riddle}@auckland.ac.nz

Abstract

Current graph-neural-network-based (GNN-based) approaches to multi-hop questions integrate clues from scattered paragraphs in an entity graph, achieving implicit reasoning by synchronous update of graph node representations using information from neighbours; this is poorly suited for explaining how clues are passed through the graph in hops. In this paper, we describe a structured Knowledge and contextual Information Fusion GNN (KIFGraph) whose explicit multi-hop graph reasoning mimics human step by step reasoning. Specifically, we first integrate clues at multiple levels of granularity (question, paragraph, sentence, entity) as nodes in the graph, connected by edges derived using structured semantic knowledge, then use a contextual encoder to obtain the initial node representations, followed by step-by-step two-stage graph reasoning that asynchronously updates node representations. Each node can be related to its neighbour nodes through fused structured knowledge and contextual information, reliably integrating their answer clues. Moreover, a masked attention mechanism (MAM) filters out noisy or redundant nodes and edges, to avoid ineffective clue propagation in graph reasoning. Experimental results show performance competitive with published models on the HotpotQA dataset.

1 Introduction

Question Answering (QA) is amongst the most commonly used tasks to quantify the reasoning and understanding ability of artificially intelligent systems. The performance of the most successful approaches on QA tasks such as SQuAD (Rajpurkar et al., 2016), TriviaQA (Joshi et al., 2017), SearchQA (Dunn et al., 2017), now far exceeds that of humans (Wang et al., 2018). However, most studies on these tasks have focused on single-hop reasoning, where most questions can be answered with a single document and without complex reasoning.

Q: In which 2015 British-American romantic drama film **directed by Todd Haynes** did **John Magaro star in**?

Selected paragraphs:

P1 Title: John Magaro
S1 **John Robert Magaro** (born February 16, 1983) is an American film, ……
S2 He starred alongside James Gandolfini in "Not Fade Away" (2012), ……
S3 **He** also **starred** alongside Rooney Mara **in** "**Carol**" (2015).

P2 Title: Carol (film)
S4 **Carol** is a 2015 British-American romantic drama film **directed by Todd Haynes**.
S5 The screenplay, written by Phyllis Nagy, is based on the 1952 romance novel ……
S6 The film stars Cate Blanchett, Rooney Mara, Sarah Paulson, Jake Lacy, and ……

P3 ……

Supporting facts: **S1**, **S4**
Answer: **Carol**

Figure 1: An example of a multi-hop question showing the utility of structured semantic knowledge for complex multi-hop reasoning. The blue dotted line denotes a coreference link between an entity *John Magaro* and a mention *He*. The green dashed lines denote semantic links between entities extracted from Open Information Extraction, *i.e., (He, star in, Carol)* and *(Carol, directed by, Todd Haynes)*.

For complex multi-hop reasoning, requiring multiple steps, these prior tasks provide a poor test. The multi-hop HotpotQA (Yang et al., 2018b) dataset, by contrast, is designed for systems that integrate information from multiple documents, reasoning to an answer explained using supporting facts.

Figure 1 is an example from HotpotQA, showing that structured knowledge, *i.e.,* co-references and RDF triples, are useful in complex multi-hop reasoning. To answer the question, the multi-hop QA model must first use the coreference in which *He* in S2 refers to *John Robert Magaro* in S1, allowing it to integrate *(He, star in, Carol)* with question-related structured knowledge *(Carol, directed by, Todd Haynes)*, thus reaching the final answer *Carol*.

Most multi-hop QA models extract entities related to the question to construct an entity graph, and then apply a GNN-based model to integrate information across nodes and predict answers (Dhingra et al., 2018; Qiu et al., 2019; Fang et al., 2020; Shao et al., 2020). However, this kind of GNN-based approach leaves challenges.

First, existing graph construction methods do

not precisely capture the semantic relationships between nodes, leading to unreliable integration of neighbouring nodes' information during graph reasoning. Figure 1 shows how essential such structured semantic knowledge is in multi-hop reasoning. Thus, integrating the semantics of input documents within GNN-based QA models remains a critical challenge. Moreover, graph reasoning over noisy or redundant nodes and edges may lead to ineffective information integration, of *e.g.,* the spurious *(He, star in, Not Fade Away)* in S1. This necessitates filtering out such "noise" nodes and edges unrelated to the question.

Second, most multi-hop QA models combine all clues related to the question into a graph, and then apply GNN-based inference to update all node representations synchronously without considering the order of clues in the reasoning chains. In this type of approach, it is difficult to explain how the models make decisions and how clues are passed through the graph in hops (Du et al., 2019).

In this paper, we propose a structure knowledge and contextual information fusion GNN for multi-hop QA. Our approach involves three steps: clue extraction, clue reasoning, and multi-task prediction. For clue extraction, we extract clues at multiple levels of granularity (question, paragraph, sentence, entity) as nodes, connected by semantic edges derived using structured knowledge. The motivation is that semantic edges provide more reliable information about neighbouring nodes for graph reasoning, compared to manually defined graph construction rules (Fang et al., 2020). For clue reasoning, inspired by step-by-step reasoning from CogQA (Ding et al., 2019), we first initialize all node representations using a pretrained contextual encoder, and then mimic human-like step-by-step reasoning via asynchronous update of node representations on the semantic graph. This update is a two-stage process in which nodes directly related to the question are updated first as direct clues, *e.g.,* entities *Todd Haynes* and *John Magaro* in Figure 1, followed by the remaining nodes which are updated as indirect clues, *e.g., Coral*. At this point, we also apply a masked attention module to filter out noisy or spurious nodes and edges to avoid ineffective or deleterious clue propagation during GNN inference. Finally, the updated node representations are passed to a multi-task layer that predicts the final answer, the answer type, supporting facts and an interpretable reasoning chain.

We evaluated our proposed KIFGraph on HotpotQA dataset and achieved a high rank amongst published systems on the leaderboard. The main contributions of this paper are as follows:

- We construct a graph based on information fusion of structured knowledge, and contextual information, at multiple levels of granularity.
- We propose applying two-stage graph reasoning for multi-hop QA, which introduces interpretability to our reasoning model via a propagation process of information from direct to indirect clues that described by the model's outputs.
- We introduce a masked attention module to filter out noisy nodes and edges to avoid ineffective clue propagation in graph reasoning.

Hence, KIFGraph[1] provides a new perspective on how to perform global interpretable reasoning through GNN-based methods, and achieves competitive performance on the HotpotQA benchmark.

2 Related work

Knowledge-based multi-hop QA. Knowledge-based QA (KBQA) usually provide accurate answers because they use reliable inference to search structured knowledge curated by humans. CNNSM (Yih et al., 2014) decomposes questions into an entity mention and a relation pattern, then maps them to the entities and relations in a knowledge base to answer a question. HSP (Zhang et al., 2019) uses a three-stage parsing architecture to generate a logical form for complex questions, and then queries an existing database to arrive at an answer. Unfortunately, KBQA is constrained by the paucity of available external knowledge bases and limited ability to use contextual information.

Question decomposition for multi-hop QA. Recent studies have focused on decomposing multi-hop questions into single-hop sub-questions, enabling existing single-hop QA models to be applied. DecompRC (Min et al., 2019) uses a pointer model to split the question and generate sub-questions, and then answers these sub-questions using an existing single-hop QA model. QDMR (Wolfson et al., 2020) trains a seq-to-seq model to parse multi-hop questions into a sequence of query steps. These QA systems attempt to find the essential

[1] https://github.com/Tswinggg/KIFGraph

clue for each sub-question, but largely ignore any relationships between the sub-questions.

Graph Neural Networks for multi-hop QA. Motivated by work with GCNs (Kipf and Welling, 2017), recent studies have proposed that one should construct entity graphs from relevant paragraphs and apply GCNs to perform implicit reasoning by propagating contextual information along graph edges. Entity-GCN (Cao et al., 2019), MHQA-GRN (Song et al., 2018), Coref-GRN (Dhingra et al., 2018) and DFGN (Qiu et al., 2019) select entity nodes and use rules to construct edges in the entity graphs. HDE-Graph (Tu et al., 2020) and HGN (Fang et al., 2020) construct a heterogeneous graph at multiple levels of granularity to maximise direct propagation of information via graph edges.

3 Methodology

In this section, we describe in overview the KIF-Graph model in Figure 2. KIFGraph involves the following three steps: i) clue extraction, including use of a paragraph retrieval module and a semantic graph construction module; ii) clue reasoning, including the masked attention and two-stage graph reasoning module at the centre of the figure; and iii) multi-task prediction, including answer-span prediction, answer-type prediction, supporting facts prediction and reasoning chain generation.

3.1 Clue Extraction

We first utilize a paragraph retriever to select paragraphs related to the question. Then, we extract multiple-granularity clues from these paragraphs as nodes, and construct semantic edges between nodes using multiple modules, in this case Named Entity Recognition (NER), Coreference Resolution (CR) and Open Information Extraction (OpenIE) (Angeli et al., 2015). This architecture allows for later extensions of the set of knowledge sources.

Paragraph Retriever. The task of the paragraph retriever is to select relevant paragraphs that contain clues related to the question Q from given input paragraphs P, where P = $\{p_0, p_1, ..., p_i, ..., p_m\}$, m is the number of paragraphs.

$$\mathrm{P_Q} = \mathrm{ParagraphRetriever(Q, P)} \tag{1}$$

The $\mathrm{P_Q}$ should contain the multiple paragraphs needed for complex multi-hop reasoning required by the question. To obtain these paragraphs, all useful clues in the question should be utilized. Similar to HGN (Fang et al., 2020), we combine a two-step hyperlink search and a paragraph ranker to retrieve relevant paragraphs from Wikipedia. The two-step hyperlink search contains two processes: i) selecting paragraphs whose title appears in the question as the first-hop paragraphs; ii) selecting second-hop paragraphs whose title appears in hyperlinks (provided by Wikipedia) in the first-hop paragraph. If this search also fails, we use the paragraph ranker, which is based on a pre-trained RoBERTa model (Liu et al., 2019), to select paragraphs with the highest ranking score.

Semantic Graph Construction. Existing studies construct graphs in GNN based on manually defined rules, *e.g.,* HGN and DFGN. In these methods, the semantic relationships, at multiple levels of granularity, between nodes have not adequately been considered, so nodes lack reliable neighbouring nodes for use during GNN node-representation updates. To provide them, we extract clues at multiple levels of granularity (question, paragraph, sentence, entity) as nodes $\mathcal{N}$, and then use structured knowledge extracted by multiple modules (*i.e.,* NER, CR and OpenIE) to generate semantic edges $\mathcal{E}$ and construct a semantic graph $\mathcal{G} = (\mathcal{N}, \mathcal{E})$. Specifically, we first use CR techniques to extract entity-mention pairs in retrieved paragraphs.

$$\begin{aligned} &\mathrm{coref}_{p_i}(\mathrm{pairs}) = \mathrm{CR(p_i)} \\ &\mathrm{pairs} = \{(\mathbf{s_e}, \mathbf{e}), (\mathbf{s_{m_1}}, \mathbf{m}_1), ..., (\mathbf{s_{m_k}}, \mathbf{m}_k)\} \end{aligned} \tag{2}$$

where $\mathrm{coref}_{p_i}(\mathrm{pairs})$ denotes the set of entity-mention pairs in paragraph p_i, $\mathbf{s}_e$ is the sentence id in which entity $\mathbf{e}$ is located, and $\mathbf{s}_{\mathbf{m}_k}$ is the sentence id of mention $\mathbf{m}_k$. Then, we iterate over the set $\mathrm{coref}_{p_i}(\mathrm{pairs})$ to build semantic edges as follows:

$$edge(\mathbf{s}_i, \mathbf{s}_j) = \begin{cases} 1, if\ \mathbf{e}\ in\ \mathbf{s}_i\ and\ \mathbf{m}_k\ in\ \mathbf{s}_j \\ 0, \qquad otherwise \end{cases} \tag{3}$$

$$edge(\mathbf{e}, \mathbf{s}_j) = \begin{cases} 1, if\ \mathbf{e}\ in\ \mathbf{s}_i\ and\ \mathbf{m}_k\ in\ \mathbf{s}_j \\ 0, \qquad otherwise \end{cases} \tag{4}$$

where if entity $\mathbf{e}$ in sentence $\mathbf{s}_i$ and mention $\mathbf{m}_k$ in sentence $\mathbf{s}_j$, it indicates that there is a semantic relationship between these two sentences. In this way, we build two types of semantic edges: $edge(\mathbf{s}_i, \mathbf{s}_j)$ between sentence nodes and $edge(\mathbf{e}, \mathbf{s}_j)$ between sentence nodes and entity nodes, *i.e.,* blue dotted lines in Figure 3.

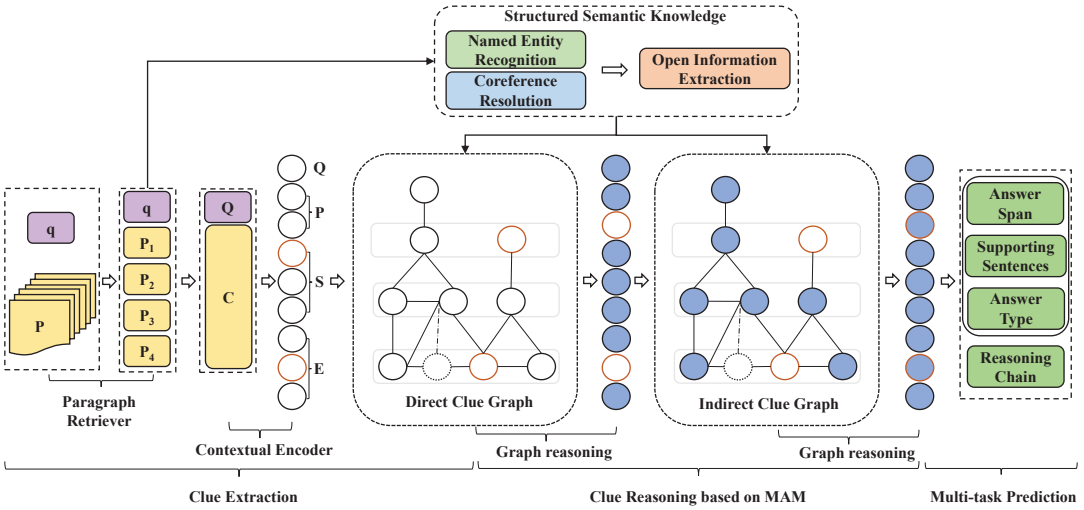

Figure 2: An overview of our proposed KIFGraph model. Specifically, it involves three key modules, i) Clue extraction extracts clues related to the question into a graph; ii) Graph reasoning based on MAM performs two-stage graph reasoning, from direct clue (white circle) graph to indirect clue (orange circle) graph, to asynchronous update node representations; iii) Multi-task prediction conducts a multi-task layer to predict the span of answer, supporting facts, the type of question and reasoning chains.

Similarly, structured knowledge between entities is important because it can represent semantic relationships between entities accurately. To obtain this, we first use OpenIE techniques to extract this structured knowledge in the form of RDF triples.

$$\text{Triple}_{p_i}(S, O, R) = \text{RDF}(p_i) \tag{5}$$

where S, O and R represent $Subject$, $Object$ and $Relationship$ respectively, and $\text{Triple}_{p_i}(\cdot)$ is the set of RDF triples in paragraph p_i. We then build semantic edges between entity nodes based on these RDF triples. Since both S and O may be a span of text, we build edges between entities as follow:

$$\text{edge}(\mathbf{e}_i, \mathbf{e}_j) = \begin{cases} 1, if\ \mathbf{e}_i\ in\ S\ and\ \mathbf{e}_j\ in\ O \\ 0, \qquad otherwise \end{cases} \tag{6}$$

To sum up, the semantic graph $\mathcal{G}$ for the example in Figure 3 consists of four types of nodes (**Q**uestion, **P**aragraph, **S**entence, **E**ntity), four direct edges built by HGN (Fang et al., 2020), *i.e.,* Q-P, P-P, P-S, S-E, and three semantic edges that we construct, *i.e.,* S-S, S-E, E-E.

3.2 Clue Reasoning

Given the semantic graph, we perform two-stage reasoning over the graph, where node representations are updated asynchronously by mimicking human step-by-step reasoning from direct clues to indirect clues. Specifically, we set node representations $\mathbf{q}, \mathbf{p}_i, \mathbf{s}_i, \mathbf{e}_i \in \mathbb{R}^d$ and graph representation $\mathbf{G} = [\mathbf{q}, \mathbf{P}, \mathbf{S}, \mathbf{E}] \in \mathbb{R}^{t \times d}$, where $\mathbf{P} = \{\mathbf{p}_1, \mathbf{p}_2, ..., \mathbf{p}_{n_1}\}$, $\mathbf{S} = \{\mathbf{s}_1, \mathbf{s}_2, ..., \mathbf{s}_{n_2}\}$, $\mathbf{E} = \{\mathbf{e}_1, \mathbf{e}_2, ..., \mathbf{e}_{n_3}\}$, n_1, n_2, n_3 denote the numbers of paragraph/sentence/entity nodes in the graph, $t = n_1 + n_2 + n_3 + 1$ is the total number of nodes, and d is the dimension of nodes.

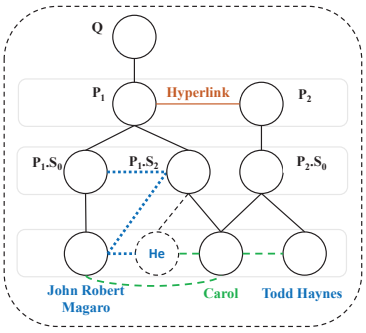

Figure 3: Semantic graph construction for the example in Figure 1. The graph consists of four types of nodes: Question Q, Paragraph P, Sentence S and Named Entity. The blue dotted lines represent semantic edges generated by coreference resolution, the green dashed lines represent semantic edges generated by OpenIE. The orange line was built using Hyperlinks as mentioned in the section Paragraph Retriever.

Contextual Encoder. We first combine all retrieved paragraphs into context C, and then initialize all representations by concatenating the question Q and the context C and feeding them into a pre-trained contextual encoder RoBERTa (Liu et al., 2019) to obtain the question representation $\mathbf{Q} = \{\mathbf{q}_0, \mathbf{q}_1, ..., \mathbf{q}_{m-1}\} \in \mathbb{R}^{m \times d}$ and the context representation $\mathbf{C} = \{\mathbf{c}_o, \mathbf{c}_1, ..., \mathbf{c}_{n-1}\} \in \mathbb{R}^{n \times d}$, where m and n are lengths of Q and C.

$$\mathbf{Q}, \mathbf{C} = \text{RoBERTa}(\text{Q}, \text{C}) \tag{7}$$

According to the previous work on Graph Attention Networks (GATs) (Velickovic et al., 2017), at

least one linear transformation is required to obtain a more expressive context representation. To this end, as an initial step in obtaining a node representation, we first apply a shared linear transformation $\mathbf{W} \in \mathbb{R}^{d\times 2d}$ to generate the higher-level context representation $\mathbf{C}' \in \mathbb{R}^{n\times 2d}$. We then apply an LSTM layer to $\mathbf{C}'$, obtaining the initial representations of all nodes $\mathbf{q}, \mathbf{p}_i, \mathbf{s}_i, \mathbf{e}_i \in \mathbb{R}^d$.

$$\begin{aligned}\mathbf{p}_i &= \mathrm{LSTM}_1(\mathbf{C}', \mathbf{p}_i^{start}, \mathbf{p}_i^{end})\\ \mathbf{s}_i &= \mathrm{LSTM}_2(\mathbf{C}', \mathbf{s}_i^{start}, \mathbf{s}_i^{end})\\ \mathbf{e}_i &= \mathrm{LSTM}_3(\mathbf{C}', \mathbf{e}_i^{start}, \mathbf{e}_i^{end})\\ \mathbf{q} &= \mathrm{MaxPooling}(\mathbf{Q})\end{aligned} \tag{8}$$

where $\mathbf{p}_i^{start}$, $\mathbf{s}_i^{start}$ and $\mathbf{e}_i^{start}$ denote the start position of the i-th paragraph, sentence and entity node, and $\mathbf{p}_i^{end}$, $\mathbf{s}_i^{end}$, and $\mathbf{e}_i^{end}$ denote the corresponding end position. MaxPooling is used to calculate the question representation $\mathbf{q}$. Finally, all node representations $\mathbf{q}, \mathbf{p}_i, \mathbf{s}_i, \mathbf{e}_i$ are concatenated as the graph representation $\mathbf{G} = [\mathbf{g_0}, \mathbf{g_1}, ..., \mathbf{g_t}] \in \mathbb{R}^{t\times d}$, t is the total number of nodes.

Masked Attention Module (MAM) Since not every node and edge contains useful information, a masked attention module penalizes spurious nodes and edges before graph reasoning. For nodes unrelated to the question, we use an attention network between question representation $\mathbf{q}$ and graph node representation $\mathbf{g}_i$ to generate the mask module $\mathbf{m}$ (Qiu et al., 2019). As a result, useful nodes related to the question will be enhanced and noisy nodes will be penalized by multiplying the mask $\mathbf{m}$ and the initial node representations $\mathbf{G}$.

$$\begin{aligned}\gamma_i &= \mathbf{q}\mathbf{W}\mathbf{g}_i\\ \mathbf{m} &= \sigma([\gamma_0, \gamma_1, ..., \gamma_i, ..., \gamma_t])\\ \mathbf{G}^\dagger &= \mathbf{m}\mathbf{G} = [m_0\mathbf{g}_0, m_1\mathbf{g}_1, ..., m_t\mathbf{g}_t]\end{aligned} \tag{9}$$

where $\mathbf{W}$ is a linear projection matrix, σ is the sigmoid function, $\mathbf{g}_i$ is the initial representation of node i and $\mathbf{G}^\dagger$ is the graph representation after masking noisy nodes.

Since GNN-based methods update node representations using information from their neighbouring nodes, noisy edges will lead to erroneous information propagation during graph reasoning. This is addressed by applying GAT to compute attention coefficients between nodes:

$$\alpha_{ij} = \frac{exp(\sigma(a^T[\mathbf{W}\mathbf{g}_i^\dagger || \mathbf{W}\mathbf{g}_j^\dagger]))}{\sum_{k\in\mathcal{G}_{i*}} exp(\sigma(a^T[\mathbf{W}\mathbf{g}_i^\dagger || \mathbf{W}\mathbf{g}_k^\dagger])} \tag{10}$$

$$\mathbf{g}_i' = \sigma(\textstyle\sum_{j\in\mathcal{G}_{i*}} \alpha_{ij}\mathbf{W}\mathbf{g}_j^\dagger) \tag{11}$$

where α_{ij} is the attention coefficient between node i and node j, $\mathbf{g}_i^\dagger \in \mathbf{G}^\dagger$ is the masked node representation of node i, $\mathbf{W} \in \mathbb{R}^{d\times d}$ is a weight matrix, σ is an activation function and $\mathcal{G}_{i*}$ are neighbours of the node i. Since the updated node representations $\mathbf{g}_i' \in \mathbf{G}'$ is a linear combination of the representations of $\mathcal{G}_{i*}$, we can also enhance useful neighbouring nodes or penalize noisy neighbouring nodes through the attention coefficient between nodes.

In summary, MAM updates the graph representation by fusing structured knowledge and contextual information, alleviating the propagation of erroneous information from noisy nodes and edges in graph reasoning and the interference with subsequent answer prediction.

Graph Reasoning. To achieve explicit and interpretable graph reasoning, we use two-stage graph reasoning based on MAM to asynchronously update all node representations according to their order in the reasoning chain, mimicking human step-by-step reasoning. Specifically, we first divide the semantic graph $\mathcal{G}$ into the direct clue graph and the indirect clue graph. The direct/indirect graph is obtained by masking out nodes that are unrelated/related to the entities in the question, *e.g.,* orange circles in the graph are direct clue nodes directly related to the question and white circles are indirect nodes in Figure 2.

$$\mathcal{G}_1, \mathcal{G}_2 = \mathrm{Divide}(\mathcal{G}) \tag{12}$$

where $\mathcal{G}_1$ and $\mathcal{G}_2$ are the direct clue graph and direct clue graph, respectively. Then, we feed the initial graph representations $\mathbf{G}$ and question representation $\mathbf{q}$ into MAM to update the representation of the direct clues, followed by the updated graph representations $\mathbf{G}'$ and question representation $\mathbf{q}'$ are passed into MAM to update the representation of the indirect clues.

$$\begin{aligned}\mathbf{G}' &= \mathrm{MAM}(\mathbf{G}, \mathcal{G}_1, \mathbf{q})\\ \mathbf{q}' &= \mathbf{G}'[0]\\ \mathbf{G}'' &= \mathrm{MAM}(\mathbf{G}', \mathcal{G}_2, \mathbf{q}')\end{aligned} \tag{13}$$

where $\mathbf{G}'$ is the graph representation of direct clue graph, $\mathbf{G}''$ is the graph representation of the overall graph. In this way, we achieve human-like step-by-step reasoning to update nodes asynchronously.

Finally, we use a gated attention mechanism (Fang et al., 2020) to merge the graph representation $\mathbf{G}''$ and the context representation $\mathbf{C}'$ for use

in the final answer prediction step.

$$\begin{aligned}\tilde{\mathbf{C}} &= \mathrm{Relu}(\mathbf{C}'\mathbf{W}_m) \cdot \mathrm{Relu}(\mathbf{G}''\mathbf{W}'_m)^T \\ \tilde{\mathbf{G}} &= \mathrm{Softmax}(\tilde{\mathbf{C}}) \cdot \mathbf{G}'' \\ \bar{\mathbf{G}} &= \sigma([\mathbf{C}'; \tilde{\mathbf{G}}]\mathbf{W}_s) \cdot \mathrm{Tanh}([\mathbf{C}'; \tilde{\mathbf{G}}]\mathbf{W}_t)\end{aligned} \tag{14}$$

where $\mathbf{W}_m \in \mathbb{R}^{2d\times 2d}$, $\mathbf{W}'_m \in \mathbb{R}^{2d\times 2d}$, $\mathbf{W}_s \in \mathbb{R}^{4d\times 4d}$ and $\mathbf{W}_t \in \mathbb{R}^{4d\times 4d}$ are trainable weight matrices. The gated representation $\bar{\mathbf{G}} \in \mathbb{R}^{4d\times 4d}$ is used for answer span prediction.

3.3 Multi-task prediction

We follow the cascade prediction module design from (Fang et al., 2020), which contains six outputs, including paragraph selection, supporting facts prediction, entity prediction, the start and end position of the answer and answer type prediction.

$$\begin{aligned}\mathbf{O}_{para} &= \mathrm{MLP}_1(\bar{\mathbf{P}}) \\ \mathbf{O}_{sent} &= \mathrm{MLP}_2(\bar{\mathbf{S}}) \\ \mathbf{O}_{entity} &= \mathrm{MLP}_3(\bar{\mathbf{E}}) \\ \mathbf{O}_{start} &= \mathrm{MLP}_4(\bar{\mathbf{G}}) \\ \mathbf{O}_{end} &= \mathrm{MLP}_5(\bar{\mathbf{G}}) \\ \mathbf{O}_{type} &= \mathrm{MLP}_6(\bar{\mathbf{G}}[0])\end{aligned} \tag{15}$$

where $\bar{\mathbf{P}}$, $\bar{\mathbf{S}}$, $\bar{\mathbf{E}}$ are updated node representations which can be obtained from $\bar{\mathbf{G}}$, $\bar{\mathbf{G}}[0]$ is the first hidden representation of $\bar{\mathbf{G}}$ and each MLP_i is a multi-layer perceptron (MLP) for different outputs.

Finally, we use cross entropy loss over each output logits. The total loss function is a weighted sum of this loss.

$$\begin{aligned}L_{total} =& L_{start} + L_{end} + \lambda_1 L_{para} + \lambda_2 L_{sent} \\ &+ \lambda_3 L_{entity} + \lambda_4 L_{type}\end{aligned} \tag{16}$$

where $\lambda_1, \lambda_2, \lambda_3$, and λ_4 are hyper-parameters, and $L_{start}, L_{end}, L_{para}, L_{sent}, L_{entity}, L_{type}$ are the cross entropy loss for the corresponding logit: $\mathbf{O}_{start}, \mathbf{O}_{end}, \mathbf{O}_{para}, \mathbf{O}_{sent}, \mathbf{O}_{entity}, \mathbf{O}_{type}$.

Finally, we select nodes in the direct clue and indirect clue graph that are not masked by MAM to generate the reasoning chain:

Q→direct clues→indirect clues→Ans

4 Experiments

In this section, we compare our system KIFGraph with state-of-the-art multi-hop QA approaches on HotpotQA (Yang et al., 2018a) dataset.

4.1 Dataset and Setup

Dataset and Metrics. We evaluate our proposed KIFGraph on HotpotQA in the distractor setting. The distractor setting contains 2 golden paragraphs and 8 distractor paragraphs. In HotoptQA dataset, there are two types of questions—Bridge question and Comparison question, and two types of answer-spans of text and yes/no. Exact Match (EM) and partial match (F1) between the prediction and the golden answer are used as performance metrics. Further, a joint metric is used to evaluate both tasks simultaneously.

Setup. In semantic graph construction phase, we use the Stanford CoreNLP toolkit (Manning et al., 2014) to extract named entities, entity-mention pairs and RDF triples from the input documents, and set the number of question/paragraph/sentence/entity nodes to 1/4/40/60. In the paragraph retrieval phase, we follow HGN to select the top-K (K=4) paragraphs for a fair comparison. In context encoding phase, the maximum input sequence of RoBERTa-large is set to 1024, the hidden layer is set to 300, the batch size is 16 and the learning rate of Adam is 1e-5. In the multi-task prediction phase, the value of $\lambda_1/\lambda_2/\lambda_3/\lambda_4$ is set to 1/1/5/1.

4.2 Main Results

In Table 1, we compare KIFGraph with other published baselines on the private test set of HotpotQA. From Table 1, we observe that KIFGraph outperforms all baselines on EM/F1 metrics of the answer and achieves the second best results on the Joint EM/F1, demonstrating the progress made by KIFGraph in answer span prediction. Compared with DFGN which constructs an entity graph and applies GATs to achieve reasoning over the entity graph, KIFGraph increase performance substantially from 59.82 to 74.12 in the Joint EM/F1 metrics. We believe this is because our model performs two-stage graph reasoning based on masked attention module, which mimics the logic of human reasoning. Compared with HGN which constructs a hierarchical graph, KIFGraph improves its answer EM/F1 by constructing a semantic graph fusing structured knowledge and contextual information. In ablation studies reported below, we provide a detailed analysis to prove that the semantic graph and two-state graph reasoning contribute to its performance.

Model	Ans		Sup		Joint	
	EM	F1	EM	F1	EM	F1
Baseline Model (Yang et al., 2018a)	45.60	59.02	20.32	64.49	10.83	40.16
DecompRC (Min et al., 2019)	55.20	69.63	-	-	-	-
OUNS (Perez et al., 2020)	66.33	79.34	-	-	-	-
DFGN (Qiu et al., 2019)	56.31	69.69	51.50	81.62	33.62	59.82
IRC (Nishida et al., 2021)	58.54	72.67	36.56	79.53	23.57	59.43
TAP2 (Glass et al., 2020)	66.64	79.82	57.21	86.69	41.21	70.65
SAE-large (Tu et al., 2020)	66.92	79.62	61.53	86.86	45.36	71.45
C2F Reader (Shao et al., 2020)	67.98	81.24	60.81	87.63	44.67	72.73
Longformer (Beltagy et al., 2020)	68.00	81.25	**63.09**	88.34	45.91	73.16
FFReader-large (Alkhaldi et al., 2021)	68.89	82.16	62.10	88.42	45.61	73.78
HGN-large (Fang et al., 2020)	69.22	82.19	62.76	**88.47**	**47.11**	**74.21**
KIFGraph	**69.53**	**82.42**	61.79	87.98	46.49	74.12

Table 1: Results on the private test of HotpotQA in the distractor setting. The proposed KIFGraph model outperforms all baselines published on the leaderboard in answer prediction. "-" denotes the case where no results are available. Leaderboard: https://hotpotqa.github.io/.

4.3 Ablation studies

In this section, we verify the effectiveness of the following three aspects of the KIFGraph model on the dev set in the distractor setting: i) The semantic graph; ii) The masked attention module; iii) The two-stage graph reasoning.

Semantic graph effectiveness. As shown in Table 2, we evaluate the impact of semantic graph by adding three different types of edges we construct. As described in the section on semantic graph construction, above, we add "sentence-sentence" and "sentence-entity" edges by CR, increasing the Ans F1 by 0.12 compared to Hier.Grpah in HGN. Adding "entity-entity" edges by OpenIE, increases the Ans F1 0.22. The combination of CR and OpenIE improves the Ans F1 by 0.28. This suggests that adding semantic edges obtained by structured knowledge is effective for multi-hop QA.

Model	Ans F1	Sup F1	Joint F1
Hier.Graph	82.22	**88.58**	74.37
Hier.Graph + CR	82.34	88.21	74.36
Hier.Graph + OpenIE	82.44	88.29	74.41
Semantic Graph	**82.50**	88.30	**74.45**

Table 2: Ablation study for semantic graph (SR) on dev set. Hier.Graph denotes a hierarchical graph. CR denotes adding edges by coreference resolution. OpenIE denotes adding edges by Open information extraction. Semantic Graph denotes adding edges by combining CR and OpenIE.

Masked attention module effectiveness. To verify the effectiveness of the masked attention module, we conduct four experiments to analyse the impact of punishing noisy nodes and edges: i) w/o masked nodes and edges: noisy nodes and edges are not penalized; ii) masked nodes: only noisy nodes are penalized; iii) masked edges: only noisy edges are penalized; iv) masked nodes and edges: noisy nodes and edges are penalized. As shown in Table 3, by penalizing noisy nodes unrelated to the question and weakening noisy edges, the Ans F1 is increased by 0.04 and 0.13, respectively. The result of "masked nodes" shows that penalizing noisy nodes unrelated to the question is helpful, but all of nodes have been filtered by clue extraction, thus improvement may not be obvious. The result of "masked edges" indicates that fusing semantic graph and contextual information to penalize noisy edges can lead to significant performance improvement. This indicates the importance of masked attention mechanism for graph reasoning.

Model	Ans F1	Sup F1	Joint F1
w/o masked nodes & edges	82.50	88.30	74.45
masked nodes	82.54	88.29	74.44
masked edges	82.63	88.32	74.52
masked nodes & edges	**82.65**	**88.34**	**74.60**

Table 3: Ablation study for masked attention module (MAM) on dev set. "masked nodes" and "masked edges" denote penalizing noisy nodes and noisy edges respectively.

Two-Stage graph reasoning. To verify the effectiveness of the two-stage graph reasoning (TS). We compare two different TS schemes with graph reasoning. In Table 4, we observe that the inverse TS (Inv TS, update node representations in order from indirect clues to direct clues) yields poor results, below the original GAT. But our proposed TS (update node representations in order from direct clues to indirect clues) that aligns with the logic of human reasoning increases the Ans F1 and Joint F1 by 0.17 and 0.19 respectively. This indicates that the two-stage graph reasoning is an effective form of explicit reasoning for multi-hop questions.

Overall ablation results of KIFGraph performances on dev set in the development set of HotpotQA are shown in Table 5. We observe that our three components improve the performance of KIF-

Q: In which 2015 British-American romantic drama film directed by Todd Haynes did John Magaro star in?

Selected Paragraphs:

P_1 Title: John Magaro
S_1 John Robert Magaro (born February 16, 1983) is an American film.
S_2 He also starred alongside Rooney Mara in "Carol" (2015).
P_2 Title: Carol (film)
S_4 Carol is a 2015 British-American romantic drama film directed by Todd Haynes.

Answer: **Carol** Supporting facts: **S_1, S_4**

Direct clues: {**Q**, **P_1**, **P_2**, **S_1**, **S_4**, Todd Haynes, John Magaro}
Indirect clues: {**S_2**, Carol}
Reasoning chain: **Q → Direct clues (S_1, S_4) → Indirect clues (S_2) → Answer**
Predicted Answer: **Carol** Predicted Supporting facts: **S_1**, **S_4**, **S_2**

Figure 4: An example of KIFGraph to answering multi-hop questions. Direct/Indirect clues are unmasked nodes in the direct/indirect clue graph. Reasoning chain is generated by the intrinsic structure of KIFGraph. In this example, we only select supporting sentences ($\mathbf{S_1}, \mathbf{S_4}, \mathbf{S_2}$) as clues for the final reasoning chain.

Model	Ans F1	Sup F1	Joint F1
KIFGraph (GAT)	82.50	88.30	74.45
KIFGraph (Inv TS)	81.92	88.23	73.89
KIFGraph (TS)	**82.67**	**88.39**	**74.64**

Table 4: Ablation study for two-stage graph reasoning on dev set. "GAT" denotes that we use GAT to update the representation of all nodes, "TS" denotes that we use two-stage graph reasoning which aligns with the logic of human reasoning to update all nodes representation. "Inv TS" is an inverted two-stage graph reasoning.

Model	Ans F1	Sup F1	Joint F1
DFGN	69.38	82.23	59.89
- Semantic Graph	74.34	84.65	66.41
- TS graph reasoning	72.49	83.14	64.76
HGN	82.22	**88.58**	74.37
- Semantic Graph	82.50	88.31	74.45
- Masked attention module	82.31	88.23	74.25
- TS graph reasoning	82.34	88.27	74.39
KIFGraph (SR)	82.50	88.30	74.45
KIFGraph (SR+MAM)	82.65	88.34	74.60
KIFGraph (SR+MAM+TS)	**82.67**	88.39	**74.64**

Table 5: Ablation study for KIFGraph on dev set. We take DFGN and HGN as the baseline model. The upper part is the model ablation results by adding different modules.The lower part is our model's final ablation results.

Graph to varying degrees. Adding our three components into baseline models (DFGN, HGN and KIFGraph), demonstrates obvious performance improvements from the semantic graph and masked attention, and the utility of fusing structured knowledge and contextual information.

4.4 Case Study

The example question in Figure 4, illustrates explicit reasoning in KIFGraph. The semantic graph construction method first extracts relevant clues at multiple levels of granularity. Then, clues nodes related to the question are selected by MAM module as direct clues, *i.e.,* {Q, P1, P2, S1, S4, Todd Haynes, John Magaro}. Next, nodes connected by semantic edges to direct clue nodes become direct clues, *i.e.,* {S2, Carol}, and two-stage graph reasoning updates their representations. Finally, using all updated graph node representations, the multi-task prediction module yields the final answer and supporting facts. We also generate an explicit reasoning chain "Question→Direct clues→Indirect clue→Answer" to demonstrate interpretable reasoning for the multi-hop question. Moreover, we found that our model provides larger supporting facts, including sentences with coreferences, to make reasoning more explainable, *i.e.,* S2. Since He in S2 refers to $John\ Magaro$ in S1, explainable multi-hop reasoning requires coreference resolution; the "gold standard" supporting facts (S1 and S4) do not suffice as explanations. Unfortunately, our extended supporting facts may slightly lower performance on the HotpotQA evaluation since they are not included in the gold-standard; further analysis may show that HotpotQA dataset changes are warranted.

5 Conclusion and Future work

In this paper, we apply explicit graph reasoning to extracted knowledge and contextual information for multi-hop reasoning. We extract clues at multiple levels of granularity relating entity nodes, and construct a semantic graph from these clues. We then combine a masked attention mechanism and two-stage graph reasoning to perform interpretable inference over the semantic graph. Experimental results on HotpotQA dataset show the effectiveness of our model. In future work, we hope to extend the range and precision of the entity relations used, and we hope to extend our model to accommodate more complex multi-hop questions with unknown number of hops and non-linear reasoning.

References

Tareq Alkhaldi, Chenhui Chu, and Sadao Kurohashi. 2021. Flexibly focusing on supporting facts, using bridge links, and jointly training specialized modules for multi-hop question answering. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 29:3216–3225.

Gabor Angeli, Melvin Jose Johnson Premkumar, and Christopher D. Manning. 2015. Leveraging linguistic structure for open domain information extraction. In *ACL*, pages 344–354, Beijing, China. Association for Computational Linguistics.

Iz Beltagy, Matthew E. Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. *CoRR*, abs/2004.05150.

Nicola De Cao, Wilker Aziz, and Ivan Titov. 2019. Question answering by reasoning across documents with graph convolutional networks. In *NAACL-HLT*, pages 2306–2317.

Bhuwan Dhingra, Qiao Jin, Zhilin Yang, William W. Cohen, and Ruslan Salakhutdinov. 2018. Neural models for reasoning over multiple mentions using coreference. In *NAACL-HLT*, pages 42–48.

Ming Ding, Chang Zhou, Qibin Chen, Hongxia Yang, and Jie Tang. 2019. Cognitive graph for multi-hop reading comprehension at scale. In *ACL*, pages 2694–2703.

Mengnan Du, Ninghao Liu, and Xia Hu. 2019. Techniques for interpretable machine learning. *Communications of the ACM*, 63(1):68–77.

Matthew Dunn, Levent Sagun, Mike Higgins, V. Ugur Güney, Volkan Cirik, and Kyunghyun Cho. 2017. Searchqa: A new q&a dataset augmented with context from a search engine. *CoRR*, abs/1704.05179.

Yuwei Fang, Siqi Sun, Zhe Gan, Rohit Pillai, Shuohang Wang, and Jingjing Liu. 2020. Hierarchical graph network for multi-hop question answering. In *EMNLP*, pages 8823–8838.

Michael Glass, Alfio Gliozzo, Rishav Chakravarti, Anthony Ferritto, Lin Pan, G P Shrivatsa Bhargav, Dinesh Garg, and Avi Sil. 2020. Span selection pre-training for question answering. In *ACL*, pages 2773–2782.

Mandar Joshi, Eunsol Choi, Daniel S. Weld, and Luke Zettlemoyer. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *ACL*, pages 1601–1611.

Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *ICLR*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692.

Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *ACL*, pages 55–60.

Sewon Min, Victor Zhong, Luke Zettlemoyer, and Hannaneh Hajishirzi. 2019. Multi-hop reading comprehension through question decomposition and rescoring. In *ACL*, pages 6097–6109.

Kosuke Nishida, Kyosuke Nishida, Itsumi Saito, and Sen Yoshida. 2021. Towards interpretable and reliable reading comprehension: A pipeline model with unanswerability prediction. In *2021 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE.

Ethan Perez, Patrick S. H. Lewis, Wen-tau Yih, Kyunghyun Cho, and Douwe Kiela. 2020. Unsupervised question decomposition for question answering. In *EMNLP*, pages 8864–8880.

Lin Qiu, Yunxuan Xiao, Yanru Qu, Hao Zhou, Lei Li, Weinan Zhang, and Yong Yu. 2019. Dynamically fused graph network for multi-hop reasoning. In *ACL*, pages 6140–6150.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100, 000+ questions for machine comprehension of text. In *EMNLP*, pages 2383–2392.

Nan Shao, Yiming Cui, Ting Liu, Shijin Wang, and Guoping Hu. 2020. Is graph structure necessary for multi-hop question answering? In *EMNLP*, pages 7187–7192.

Linfeng Song, Zhiguo Wang, Mo Yu, Yue Zhang, Radu Florian, and Daniel Gildea. 2018. Exploring graph-structured passage representation for multi-hop reading comprehension with graph neural networks. *CoRR*, abs/1809.02040.

Ming Tu, Kevin Huang, Guangtao Wang, Jing Huang, Xiaodong He, and Bowen Zhou. 2020. Select, answer and explain: Interpretable multi-hop reading comprehension over multiple documents. In *AAAI*, pages 9073–9080.

Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2017. Graph attention networks. *CoRR*, abs/1710.10903.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *EMNLP*, pages 353–355.

Tomer Wolfson, Mor Geva, Ankit Gupta, Yoav Goldberg, Matt Gardner, Daniel Deutch, and Jonathan Berant. 2020. Break it down: A question understanding benchmark. *Trans. Assoc. Comput. Linguistics*, 8:183–198.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018a. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *ACL*, pages 2369–2380.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018b. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *EMNLP*, pages 2369–2380.

Wen-tau Yih, Xiaodong He, and Christopher Meek. 2014. Semantic parsing for single-relation question answering. In *ACL*, pages 643–648.

Haoyu Zhang, Jingjing Cai, Jianjun Xu, and Ji Wang. 2019. Complex question decomposition for semantic parsing. In *ACL*, pages 4477–4486.